Lab4实验报告

一、猜解思路

1.task1

源代码如下：

```
1110010000001110
0101000000100000
010010000000000x
1111000000100101
0111111010000000
000101001010x001
0001000000100001
0010001000010001
0001001x01111111
0011001000001111
0000010000000001
0100111111111000
0001010010111111
01x0111010000000
1100000111000000
```

step1:有四处需要填值，先给将每一条翻译为汇编语言，加入一些文字描述，如下

```
1110 010 000001110    ;LED R2 #14
0101 000 000 1 00000  ;AND R0 R0 #0 将R0清零
0100 1 00000000001    ;JSR 1
1111 000000100101     ;HALT
0111 111 010 000000   ;STR R7 base(R2)     label2
0001 010 010 1 00001  ;R2 = R2 + 1?
0001 000 000 1 00001  ;R0 = R0 + 1
0010 001 000010001    ;LD R1 #5
0001 001 001 111111   ;R1 = R1 - 1?
0011 001 000001111    ;ST R1 to label1
0000 010 000000001    ;BRz
0100 1 11111111000    ;JSR label2
0001 010 010 111111   ;R2 = R2 - 1
0110 111 010 000000   ;LDR R7 = base(R2)
1100 000 111 000000   ;RET
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000101      ;label1
```

step2:逐个猜解

- 第一个含有x的是JSR指令，x一定为1，不然x为0，跳了和不跳一样，不合逻辑;
- 第二个含有x的是加法指令，R2 = R2 + ?，若x为1则R2 = R2 + 9，x 为0，R2 = R2 + 1，待定;
- 第三个含x的是加法指令，R1 = 寄存器？ - 1，若x为1，寄存器为R5，但前面没有用R5，不合逻辑，x 为0;
- 第四个x含于指令前四位，若x 为0，操作码为JSRR显然不合逻辑，则x = 1;

step3:继续观察代码与所给结果

```
R0 = 5, R1 = 0, R2 = 300f, R3 = 0
R4 = 0, R5 = 0, R6 = 0, R7 = 3003
```

第二个x只能为0，否则结果不正确。实际运行也证明第二个x为0;

2.task2

源代码如下:

```
0010001000010101
0100100000001000
0101010001100111
0001001010000100
00010000xxx11001
00000011xxx11011
00010000xxx11001
0000100000000001
0001001001111001
1111000000100101
0101010010100000
0101011011100000
0101100100100000
0001010010100001
0001011011101000
0101101011000001
0000010000000001
0001100010000100
0001010010000010
0001xxx011000011
0000xxx111111010
1100000111000000
0000000100100000
```

step1:对机器码进行翻译，并对代码结构大致掌握

```
0010 001 000010101    ;LD R1 pcoffset #21,即去23行取数
0100 1 00000001000    ;JSR pcoffset #8
0101 010 001 1 00111  ;取R1的低三位放入R2
0001 001 010 000 100  ;R1 = R2 + R4
0001 000 0xx x 11001  ;ADD
0000 001 1xxx11011    ;结果为正数，跳转，盲猜111
0001 000 0xx x 11001  ;ADD
0000 100 000000001    ;结果为负数跳过下一条指令
0001 001 001 1 11001  ;R1 = R1 - 7
1111 0000 00100101    ;HALT
0101 010 010 1 00000  ;R2清零
0101 011 011 1 00000  ;R3清零
0101 100 100 1 00000  ;R4清零
0001 010 010 1 00001  ;R2加1
0001 011 011 1 01000  ;R3加8
0101 101 011 000 001  ;R5 = R3 & R1
0000 010 000000001    ;结果为0，跳过下一条指令
0001 100010000100
0001 010 010 000 010  ;R2 = 2*R2
0001 xxx 011 000 011  ;? = 2*R3
0000 xxx 111111010    ;结果为？，跳回16
1100 000 111 000000   ;RET
0000 000100100000
```

step2:以xxx为一个基本模块进行逐个猜解

- 第一和第三个出现于加法指令，且上面一条指令对R1操作，猜两个xxx均为011；
- 第二个出现在BR指令的PCoffset中，考虑跳转最可能在该程序中，所以xxx为111；
- 第四个结合提示“利用了除8”，且位于一个循环中，应该是将R3自身左移，xxx为011；
- 第五个是一个出口设计，应该是当R3为0时跳出循环，所以不为0时因继续循环，所以xxx = 101；

step3:综合考虑，验证猜想

该程序目的是求除7的余数，代码大体框架为一个主程序，一个子程序，子程序实现功能为将R1原本的值右移3位。主程序通过算法得到余数。运行后验证猜想正确。

二、实现代码

task1:

```
;开始程序
0011 0000 0000 0000
;
1110 010 000001110    ;R2 <= x300f
0101 000 000 1 00000  ;将R0清零
0100 1 00000000001    ;跳到子程序
1111 000000100101     ;HALT
;label1
0111 111 010 000000   ;R2指定的地址 <= R7
0001 010 010 1 00001  ;R2 = R2 + 1?
0001 000 000 1 00001  ;R0 = R0 + 1
;5次循环
0010 001 000010001    ;R1 <= MDR,初始值为5
0001 001 001 111111   ;R1 = R1 - 1?
0011 001 000001111    ;MDR <= R1
0000 010 000000001    ;所得结果为0跳过下一行
0100 1 11111111000    ;跳回子程序开头
;
0001 010 010 111111   ;R2减1
0110 111 010 000000   ;R7 <= R2
1100 000 111 000000   ;RET
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000101
```

task2:

```
0011 0000 0000 0000
0010 001 000010101    ;去最后取数放到R1中
;lable1
0100 1 00000001000    ;跳到子程序
0101 010 001 1 00111  ;取R1的低三位放入R2
0001 001 010 000 100  ;R1 = R2 + R4
0001 000 001 1 11001  ;R1减7放入R0中
0000 001 111111011    ;结果为正数，跳到子程序
```

```
0001 000 001 1 11001  ;R1减7放入R0中
0000 100 000000001    ;结果为负数跳过下一条指令
0001 001 001 1 11001  ;R1 = R1 - 7
1111 0000 00100101    ;HALT
;label2
;整体3结果将该数右移三位
0101 010 010 1 00000  ;R2清零
0101 011 011 1 00000  ;R3清零
0101 100 100 1 00000  ;R4清零
0001 010 010 1 00001  ;R2 <= 4'b0001
0001 011 011 1 01000  ;R3 <= 4'b1000
0101 101 011 000 001  ;R5 = R3 & R1
0000 010 000000001    ;结果为0，跳过下一条指令
0001 100 010 000 100  ;R4 = R4 + R2
0001 010 010 000 010  ;R2 << 1
0001 011 011 000 011  ;R3 << 1
0000 101 111111010    ;结果不为0，跳回20
1100 000 111 000000   ;RET
;label3
0000 000100100000
```

三、思考

1.阅读代码和写代码能力同等重要。

2.好的算法带来效率的飞升。